Using the Attunity Connect Syntax File

Version 4.1



Using the Attunity Connect Syntax File

© 2003 by Attunity Ltd.

Due to a policy of continuous development, Attunity Ltd. reserves the right to alter, without prior notice, the specifications and descriptions outlined in this document. No part of this document shall be deemed to be part of any contract or warranty whatsoever.

Attunity Ltd. retains the sole proprietary rights to all information contained in this document. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopy, recording, or otherwise, without prior written permission of Attunity Ltd. or its duly appointed authorized representatives.

Product names mentioned in this document are for identification purposes only and may be trademarks or registered trademarks of their respective companies.

3rd party software credits

This product (Attunity Connect) includes software developed by Eclipse.org, Exolab.org, Sun Microsystems, Inc., the JDOM project (http://www.jdom.org/) and the Apache Software Foundation (http://www.apache.org/).

Attunity hereby disclaims on behalf of all Eclipse.org contributors whose components are included in Attunity Connect, all warranties and conditions, express and implied, including warranties or conditions of title and non-infringement, and implied warranties or conditions of merchantability. Attunity excludes on behalf of all Eclipse.org contributors whose components are included in Attunity Connect all liability for damages, including direct, indirect, special, incidental and consequential damages.

Attunity hereby agrees to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software, as related to Sun's software components embedded in this software (Attunity Connect).

Castor is Copyright 2000-2002 (C) Intalio Inc. All Rights Reserved.

JDOM is Copyright (C) 2000-2002 Brett McLaughlin & Jason Hunter. All rights reserved.

The Apache Software License, Version 1.1 is Copyright (c) 1999-2000 The Apache Software Foundation. All rights reserved.

Table of Contents

Handling Different Flavors of SQL	5
The Syntax File	6
Defining a Flavor of SQL	9
Specifying Variations in the Syntax File	10
Variations in SQL Statements	10
Supported Functionality	11
Variations in SQL Functions	
Operands Supported by a Function	
Date and Time Constants	
Example Syntax File Definitions	
Default SQL Flavors Supplied by Attunity Connect	19
Specifying a Flavor of SQL in an Application Connect String	20

iv

Handling Different Flavors of SQL

Attunity Connect processes the SQL submitted by a user based on the backend database being accessed. In the following circumstances you can control the way the SQL is processed by Attunity Connect:

- When the features supported by the version of the backend database are different from the support provided by Attunity Connect for that database.
- When the backend database is accessed using either the ODBC or OLESQL Attunity Connect generic drivers. The set of SQL features sent by default to the backend database is minimal those that are normally supported by all relational databases. This is because any flavor of SQL can be supported by the backend database.

Attunity Connect provides a mechanism to handle these situations, using a special file (the Attunity Connect SQL syntax file).

The following topics are covered:

The Syntax File – For details, see page 6.

Defining a Flavor of SQL – For details, see page 9. This section includes the following:

Specifying Variations in the Syntax File - This section includes the following:

Variations in SQL Statements – For details, see page 10.

Variations in SQL Functions – For details, see page 14.

Supported Functionality – For details, see page 11.

Operands Supported by a Function – For details, see page 15.

Default SQL Flavors Supplied by Attunity Connect – For details, see page 19.

Specifying a Flavor of SQL in an Application Connect String - For details, see page 20.

The Syntax File

The Attunity Connect SQL syntax file is a text file defining the SQL flavor specific to relational providers having an SQL syntax other than those supported by default by Attunity Connect. This file (NAV.SYN or NAVSYN on Compaq NonStop Guardian and IBM MVS platforms) resides in the DEF directory under the directory where Attunity Connect is installed (on Compaq NonStop platforms, in the subvolume where Attunity Connect is installed, and on IBM MVS as NAVROOT.DEF.NAVSYN, where NAVROOT is the high level qualifier where Attunity Connect is installed).

A syntax file resides on every Attunity Connect machine.

The syntax file is divided into sections. Each section defines a different flavor of SQL or modifies an existing one. The syntax file shipped with Attunity Connect contains the following predefined flavors:

SQL Flavor	Syntax Name in Syntax File
Oracle case-sensitive data	Oracle version 8 - ORACLE8_SYNTAX Oracle version 7 - ORACLE_SYNTAX With this syntax use quotes (") to delimit the name for case sensitive table and columns names. For example: SELECT * FROM "Employee" WHERE "Employee-Status" = 'Married' You must specify precisely the case sensitivity of the names.
RdbSQL version 5	RDB5_SYNTAX
OLESQL driver against JOLT	OLESQL_JOLT
ODBC driver against SQL/MX data	SQLMX_SYNTAX
ODBC driver against SYBASE SQL AnyWhere data	SQLANY5_SYNTAX
ODBC driver against EXCEL data	excel_syntax ^a

a. The syntax defines the single quote in Excel as the `character instead of the usual 'character.

The following flavors of SQL are supported by Attunity Connect without the need for a syntax file entry. These flavors can be used as a base for a new flavor in the syntax file and they can be also specified in the binding file directly as a flavor for a database:

SQL Flavor	Attunity Connect Syntax Name
DB2 under OS/390	DB2MF
DB2 under UNIX and NT	DB2
Informix	INFORMIX
Ingres	INGRES
Ingres II ^a	OPENINGRES
Microsoft JET driver-based data (such as MS Access) ^b	OLESQL_JET ^c
ODBC-based data	ODBC
OLE DB-based data	OLESQL
ORACLE version 7	ORACLE
ORACLE version 8 and higher	ORACLE8
RdbSQL version 6	RDBSQL
Red Brick Warehouse	REDBRICK
SQL Server	MSSQL
SQL/MP	SQLMP
Sybase	SYBASE

- a. Use the Ingres driver.
- b. Use either the ODBC or OLESQL driver.
- c. The OLESQL_JET syntax includes support for data with table or column names that contain spaces.
- When you access a database through any of the drivers supplied by Attunity Connect (such as Informix or Sybase), the SQL flavor of that database is supported automatically.

The Attunity Connect syntax file enables you to do the following:

- Define a new flavor of SQL.
- Modify a flavor of SQL implemented by Attunity Connect.

Syntax File Format for a New Flavor of SQL

A syntax file section that specifies a new flavor of SQL has the following format:

```
[syntax_name]
BASED_ON = syntax_name_of_base_flavor
variation1
variation2
```

On an IBM MVS platform, the sections use angled brackets (<s yntax_name>).

TDP_TYPE can be used as a synonym for BASED_ON.

where:

syntax_name – The name of a new flavor. This name must be unique.

syntax_name_of_base_flavor – The name of an existing flavor that serves as the basis of your flavor. If your omit this parameter, the new SQL flavor is based on the Attunity ConnectSQL syntax. The name can be one of the flavors previously defined in the syntax file (including the predefined flavors listed in the table shown on page 6) or one of the flavors supplied by Attunity Connect that do not require a syntax file entry (listed in the table shown on page 7).

variation – The specific variations in your custom SQL (see "Specifying Variations in the Syntax File" on page 10).

Syntax File Format for a Modified Flavor of SQL

Modifying a flavor of SQL supported by Attunity Connect will change the SQL of all databases using this flavor. The syntax file section has the following format:

```
[syntax_name]
variation1
variation2
```

On an IBM MVS platform, the sections use angled brackets (<syntax_name>).

where:

syntax_name – The name of an existing SQL flavor provided by Attunity Connect. The name can be one of the flavors previously defined in the

syntax file (including the predefined flavors listed in the table shown on page 6) or one of the flavors supplied by Attunity Connect that do not require a syntax file entry (listed in the table shown on page 7).

The syntax file must be changed to reflect the modified syntax on every machine where the flavor is used (including the client machine).

variation – The specific variations in your custom SQL (see "Specifying Variations in the Syntax File" on page 10).

Ordering Sections in the Syntax File

The order of the sections in the syntax file is important. For example, if you have a section modifying an existing flavor (such as ORACLE) and afterwards a section defining a new flavor (such as MY_ORACLE) based on ORACLE, changes in the former section affect MY_ORACLE flavor. However, if the MY_ORACLE section appears first in the syntax file, it is based on the original flavor for ORACLE supplied by Attunity Connect.

Errors in the Syntax File

When you execute a query that uses an SQL flavor defined in the Attunity Connect SQL syntax file, Attunity Connect checks that the syntax defined for that data source has been correctly defined. If a mistake is found, an error is issued and that syntax section in the syntax file is ignored by Attunity Connect.

Defining a Flavor of SQL

To specify a flavor of SQL to be used by Attunity Connect for a specific database, you must define the flavor in the syntax file (or use one of the predefined flavors) and change the binding file to reference this flavor. You can make these changes directly to the files or in the application connect string (see page 20 for details about defining an SQL flavor in a connect string).

To define a flavor of SQL to Attunity Connect:

- 1. If you want a flavor not previously supported by Attunity Connect, on every machine where you want a specific SQL flavor to be used (including the client machine), specify the SQL flavor in the syntax file using a text editor, as described in "The Syntax File" on page 6.
 - When the SQL flavor of your database is the same as one of the flavors of SQL supported by Attunity Connect (either a flavor previously defined in the syntax file, including the predefined flavors listed in the table shown on page 6, or one of the flavors supplied by Attunity Connect that do not require a syntax file, listed in the table shown on page 7), continue with the next step.
- 2. In the Data Access Setup wizard, in the Advanced Database Properties screen for the relevant data source, specify the **Syntax**

name. The Syntax name identifies the syntax_name section you specified in the syntax file.

This generates the syntaxName parameter in the binding file, as follows:

```
<datasource name="name" type="type"</pre>
            connect="connect string"
             syntaxName="syntax name" />
```

Only the first 15 characters of the syntaxName are used.

Specifying Variations in the Syntax File

Attunity Connect enables you to specify the following types of variations of standard SQL flavor:

- Variations in SQL Statements (see page 10).
- Supported Functionality (see page 11).
- Variations in SQL Functions (see page 14).
- Operands Supported by a Function (see page 15).
- Date and Time Constants (see page 17).

the Syntax File

Using Single Quotes in A single quote is reserved by Attunity Connect to identify parameters. If you need to specify a single quote (') as part of an entry in the syntax file, use two single quotes (").

Variations in SQL Statements

A variation in an SQL statement has the following format:

variation = value

where Attunity Connect enables you to specify the following SQL variations:

	Parameter in Syntax File	Max. Length
Syntax for a column alias.	ASFORM	5
The string used to denote a parameter value. You can incorporate %d into the string that will be replaced by a sequential counter of parameters (for example, if your driver supports parameters of the form:p1,:p2,, specify ":p%d").	PLACEHOLDERS	5

	Parameter in Syntax File	Max. Length
The character used to quote identifier and table names. If this value is not specified, there is no quoting of identifier or table names.	IDENTIFIER_QUOTE_CHAR	1
The character used to quote a character string. If this value is not specified, there is no quoting of character strings.	STRINGQUOTATION	1
The join connector used between tables in a FROM clause.	JOINCONNECTOR	20
Starting and ending characters used to group two tables in a FROM clause, such as an open parenthesis "(" at the beginning and a close parenthesis ")" at the end.	JOINSURROUNDCHARS	2

Supported Functionality

You can specify whether the database you want to access supports a particular SQL functional variation. The following parameters denote the functional variations Attunity Connect supports. Specifying Y (yes) for a parameter indicates that this functional variation is supported by the database; specifying N (no) indicates that the database does not support the functional variation.

AGG_DIST_NOT_SUPPORT – The database doesn't support DISTINCT in an aggregate function.

CASE_SENSITIVE – The database is case sensitive. When set to YES, identifiers are passed to the backend database exactly as they are specified in the SQL passed to Attunity Connect. When set to NO, identifiers are converted to upper case before being sent to the backend database. When set to CASE_SENSITIVE_QUOTE, identifiers are passed to the backend database as they are written only if they are surrounded by quotes, otherwise they are converted to upper case.

For Oracle data, set the value to CASE_SENSITIVE_QUOTE and surround the case sensitive table and column names in quotes.

EXPR_IN_ORDERBY – The database supports expressions in the ORDER BY list. An integer constant is treated as an ordinal. This parameter is ignored if ORDER_NO_CALC is specified.

If an ORDER BY clause is not specified, ORDER BY num is allowed, where num can refer to any expression in the list of columns retrieved by the SELECT statement.

Also see: ORDER_BY_COLUMN, ORDER_EXPR_BY_NUM and ORDER_NO_CALC.

GROUP ANY EXPR – Any expression can appear in a GROUP BY clause.

If GROUP ANY EXPR is specified, GROUP BY COLUMN and GROUP_EXPR_BY_NUM are ignored.

GROUP_BY_ALIAS – 'GROUP BY *alias*' syntax is allowed, where *alias* is an alias assigned to a column in the list of columns retrieved by the SELECT statement

GROUP BY COLUMN – 'GROUP BY *column-name*' syntax is allowed, where *column-name* does not have to appear in the list of columns retrieved by the SELECT statement.

GROUP_BY_COLUMN can be specified with GROUP_EXPR_BY_NUM in a query. If GROUP_ANY_EXPR is specified, GROUP_BY_COLUMN is ignored.

GROUP_EXPR_BY_NUM – 'GROUP BY *num*' syntax is allowed, where num can refer to an expression.

GROUP_EXPR_BY_NUM can be specified with GROUP_BY_COLUMN in a query. If GROUP ANY EXPR is specified, GROUP EXPR BY NUM is ianored.

GROUP_SHOULD_AGG – At least one aggregate function must appear in the SELECT statement when using a GROUP BY clause. For example, "SELECT sal FROM sal GROUP BY sal" is invalid.

NO_COLUMN_ALIAS – Aliases for columns are not supported.

NO_DISTINCT_HAVING – The database doesn't support the DISTINCT operator in aggregated HAVING clauses.

NO_EXPRESSIONS_IN_INSERT – The database doesn't support expressions as part of a VALUES clause in an INSERT statement. For example, a statement like "INSERT INTO VALUES(10+10)" is not supported.

NO_GRANT – The database doesn't support user permissions within the SQL.

NO_MULTI_THREADING – The database cannot handle multithreading of the same statement or command.

NO_OWNER – The database doesn't support owner specification within the SQL.

NO_PARAMETERS_IN_HAVING - The database doesn't support parameters in a HAVING clause.

NO_PARAMETERS_IN_SUBQUERY - The database doesn't support parameters in a subquery.

NO TWO PARAMS IN COMPARE - Only one side of a comparison can be a parameter.

Also see: NO_TWO_PARAMS_IN_MATH and PARAM AND EXPR IS PARAM.

NO TWO PARAMS IN MATH - Only one side of a function involving a +, -, * or / arithmetic operator can be a parameter.

Also see: NO TWO PARAMS IN COMPARE and PARAM_AND_EXPR_IS_PARAM.

ONE COLUMN SUBQUERY – Subqueries cannot contain more than one column. An aggregate subquery with more than one column cannot be passed to the database.

ORDER_BY_COLUMN - 'ORDER BY column-name' syntax is allowed, where *column-name* does not have to appear in the list of columns retrieved by the SELECT statement.

Also see: EXPR_IN_ORDERBY, ORDER_EXPR_BY_NUM and ORDER NO CALC.

ORDER_EXPR_BY_NUM - 'ORDER BY num' syntax is allowed, where num can refer to an expression.

Also see: EXPR_IN_ORDERBY, ORDER_BY_COLUMN and ORDER NO CALC.

ORDER_NO_CALC – Ordering by a constant field is not supported.

Also see: EXPR_IN_ORDERBY, ORDER_BY_COLUMN and ORDER_EXPR_BY_NUM.

PARAM_AND_EXPR_IS_PARAM – A function that does not accept parameters also will not accept an expression that contains a parameter, even if the expression also contains at least one identifier or constant.

Also see: NO_TWO_PARAMS_IN_COMPARE and NO_TWO_PARAMS_IN_MATH.

Variations in SQL Functions

You may need to specify variations of standard SQL functions in the following cases:

- The database you want to access does not implement certain functions.
- The database you want to access implements functions using a name different from that used by Attunity Connect.
- The database you want to access implements functions using an order of the function operands different from the order used by Attunity Connect.

A variation in an SQL function has the following format:

function = rule

where:

function – Attunity Connect enables you to specify variations for the following functions:

■ ABS	LENGTH	RPAD
ASCII	■ LIKE (with 2 or 3	ROUND
■ CEIL	operands)	RTRIM
■ CASE	■ LOG10	■ SIGN
 CASE_COND^A 	LOWER	■ SQRT
■ CHR	■ LN	■ SUBSTR (with 2 or 3
CONCAT	■ LPAD	operands)
CONVERT	LTRIM	trig - trig is one of:
■ EXP	■ MOD	SIN, ASIN, SINH
■ FLOOR	■ PI	COS, ACOS, COSH
■ IFNULLL	POSITION	TAN, ATAN, TANH
■ IN (list of values)	■ POWER	■ TRUNC

a. The conditional case function.

For details about these functions see the *Attunity Connect Data Adapter Guide and Reference*.

rule – Use the tilde symbol (\sim) to represent a parameter. If you need to reference specific operands in an order different from the order in which they are passed, use '0 for first parameter, '1 for the second parameter, etc. See the example below.

Specify NOT_SUPP for any function that is not supported by your database.

In most cases, the function name alone identifies the function itself. However, in the case of the LIKE and SUBSTR functions, the function specification can also include the number of the function's operands. If either of these functions is not supported with a particular number of operands, add (2) or (3) to identify which form of the function is not supported (the form having 2 operands or the form having 3 operands). For example, the following specifies that the LIKE function with 3 operands is not implemented by the database:

```
LIKE(3) = NOT SUPP
```

The LIKE function with 2 operands, however, is supported.

Example

The database of this example has the following characteristics:

- It does not implement the LIKE function for three operands.
- It does implement the substring function with three operands, but calls it SUBSTRING (instead of the default SUBSTR).
- It does not implement the substring function with two operands. However, it does implement the LENGTH function, and you still want to pass a substring with two operands.

```
[MY-SYNTAX]
LIKE(3) = NOT_SUPP
SUBSTR = SUBSTRING(~, ~, ~)
  /* three operands ( SUBSTRING with three operands */
SUBSTR=SUBSTRING('0,'1, LENGTH('0)-'1+1)
  /* two operands ( SUBSTRING with two operands */
```

Operands Supported by a Function

You can specify the way a database handles the operands that are accepted by a function. By default, a function will accept any type of operands, however, some databases restrict the type of operands that can be accepted.

An operand in Attunity Connect is categorized as one of the following:

- Constant
- Expression
- Identifier
- Parameter

You can specify the type of operands that are supported by a function in one of the following ways:

Specify the operands that are supported (assuming initially that none are supported).

The format to specify operands that are supported is:

OPERANDS_function SUPPORT [(oper1,oper2) [(oper1,oper2) [...]]]

Each pair of operands that you specify – (oper1, oper2) – dictates a pair of operands that are accepted by the function. You must specify pairs of operands.

Specify the operands that aren't supported (assuming initially that all are supported)

The format to specify operands that are not supported is:

OPERANDS function NO SUPPORT rule rule1 ... rulen

Where rule is: $(1, oper1) \mid (2, oper2) \mid (oper1, oper2)$

(1, *oper1*) specifies that the first operand cannot be of type *oper1*.

(2, oper2) specifies that the second operand cannot be of type oper2.

(*oper1*, *oper2*) specifies that if the first operand is of type *oper1*, the second operand cannot be of type be *oper2* and if the second operand is of type *oper2*, the first operand cannot be of type be *oper1*.

where:

function – The name of the function (such as LIKE). See "Variations in SQL Functions" on page 14 for the complete list of functions that can be specified in the syntax file.

oper – The type of operand. The type can be one of the following:

CONST – A constant

EXPR – An expression

ID – An identifier

PARAM – A parameter

When the function that is specified accepts more than two operands, the specification for the second operand is applied to the all the operands except for the first operand.

When the function accepts only one operand, any values specified for a second operand are ignored.

Examples

The LIKE function does not accept a parameter for its first operand:

```
OPERANDS LIKE NO SUPPORT (1, PARAM)
```

■ The MOD function does not accept a parameter for both its operands (one operand can be a parameter, as long as the other operand is not a parameter):

```
OPERANDS MOD NO SUPPORT (PARAM, PARAM)
```

The SUBSTR function accepts only an identifier for its first operand and either an identifier or parameter for its second command:

```
OPERANDS SUBSTR SUPPORT (ID, ID) (ID, PARAM)
```

The LIKE function doesn't accept identifiers or parameters for both its first and second operands:

```
OPERANDS LIKE NO SUPPORT
       (1, ID) (1, PARAM) (2, ID) (2, PARAM)
```

Date and Time Constants

The format of the Attunity Connect date and time constants sent to the driver can be overridden.

The following is the syntax Attunity Connect supports for date and time constants:

- The DATE_CONSTANT function is defined using parameters of year (YYYY), month (MM), and day (DD).
- The TIME_CONSTANT function is defined using parameters of hours (HH) using a 24 hour clock, minutes (MI) and seconds (SS)
- The TIMESTAMP_CONSTANT function is defined using the combined parameters of DATE_CONSTANT and TIME_CONSTANT, where the first set of parameters define the DATE and the second set of parameters define the TIME.

Example Syntax File Definitions

When your database uses a flavor of SQL similar to one of the flavors of SQL supported by Attunity Connect or previously defined in the syntax file, you can use this SQL flavor as the basis for the SQL for your database.

To create a new SQL flavor based on an existing flavor:

- 1. Open the Attunity Connect SQL syntax file in a text editor on the machine where the data source is located.
- 2. In the syntax file, specify a new section:

```
[syntax name]
BASED ON = sql source
```

where:

syntax_name – A name to identify the SQL flavor (up to a maximum of 15).

If this name is the same as one of the existing flavors supported by Attunity Connect, it will replace that flavor. In this case, the syntax file must be changed on every machine where the flavor is used.

sql_source – The data source type whose SQL is similar to that of your database.

- You can base a provider's flavor on the default Attunity Connect SQL flavor (by not specifying BASED_ON = sql_source).
- 3. In this section of the syntax file, specify the variations of SQL syntax specific to the database. For a description of the available variations, see page 10.
- 4. In the Data Access Setup wizard, in the Advanced Database Properties screen, specify a **Syntax name**. The Syntax name identifies the SQL flavor in the Attunity Connect syntax file.

Example

The following code specifies an SQL flavor called OLESQL_SPECIAL. The SQL implementation has the following characteristics:

```
[OLESQL SPECIAL]
BASED ON = OLESQL /* my flavor is similar to that of
OLESQL, but ... */
 (---- specify variations here
```

The SQL implementation of this database has the following characteristics:

It is based on the OLESQL flavor provided with Attunity Connect.

- It calls the substring function SUBSTRING (rather than the default SUBSTR).
- It does not implement the substring function with 2 operands, but since it does implement the LENGTH function there is workaround for this limitation.

```
[OLESQL SPECIAL]
BASED ON = OLESQL
/* SUBSTRING with three operands */
SUBSTR = SUBSTRING(~, ~, ~)
/* SUBSTRING with two operands */
SUBSTR=SUBSTRING('0,'1, LENGTH('0)-'1+1)
```

Default SQL Flavors Supplied by Attunity Connect

Attunity Connect SQL The Attunity Connect default SQL is based on ANSI '92 SQL with the following definitions:

```
ABS = ABS(\sim)
ASFORM = "AS"
CONCAT = ~ | ~
LIMIT ROWS SYNTAX = NOT SUPP
GROUP BY COLUMN = yes
ORDER EXPR BY NUM = yes
IDENTIFIER QUOTE CHAR =
IFNULL = IFNULL(~, ~)
JOINCONNECTOR = ","
JOINSURROUNDCHARS = ""
LENGTH = LENGTH(~)
LIKE = ~ LIKE ~
LIKE = ~ LIKE ~ ESCAPE ~
LOWER = LOWER(~)
MOD = \sim MOD \sim
PLACEHOLDERS = "?"
POSITION = POSITION(~ IN ~)
SQRT = SQRT(\sim)
STRINGQUOTATION = 
SUBSTR = SUBSTR(\sim, \sim)
SUBSTR = SUBSTR (\sim, \sim, \sim)
UPPER = UPPER(~)
```

The values of the other parameters, described in "Supported Functionality" on page 11, is **n** (no).

Refer to the nav.syn file in the DEF directory, under the directory where Attunity Connect is installed for the syntaxes provided with Attunity Connect.

Specifying a Flavor of SQL in an Application Connect String

Instead of specifying a flavor of SQL in the syntax file, you can specify it directly in the connect string of your (ADO, ODBC, or JDBC) application. You define the flavor by specifying the Syntax parameter in the application connect string, as follows:

Syntax=(syn-name[,"prop=value"[,"prop=value"]...])

syn-name – A name that identifies a particular SQL flavor. If a syn-name is used which is greater than 15 characters, only the first 15 characters are used.

prop – Syntax properties. For details see "Specifying Variations in the Syntax File" on page 10.

Use a pair of double quotes (") when the specific flavor requires double quotes ("). For example, a null string in an expression ("") is specified as

Example

Assume a driver with the following characteristics:

- The driver implements the substring function with three operands, but calls it SUBSTRING (instead of the default SUBSTR).
- The driver does not implement the substring function with two operands. However it does implement the LENGTH function which you use to enable a substring function to operate with only two operands.

The Syntax entry in the connect string can be specified as follows:

```
Syntax=(MY-SYNTAX, "SUBSTR=SUBSTRING(~, ~, ~) ",
          "SUBSTR=SUBSTRING('0,'1, LENGTH('0)-'1+1)")
```

In addition, a binding entry must be specified. You can do this in one of the following ways:

By specifying the Binding parameter in the connect string, as in the following example:

```
Binding=(MYDB,ODBC,MYDSN,TDP_SYNTAX_NAME=MY-SYNTAX)
```

To specify more than one syntax section, repeat this parameter.

- TDP_SYNTAX_NAME is an Attunity Connect name for the syntaxName parameter in the binding file.
- By specifying the syntax name in the Advanced Database Properties screen of the Data Access Setup wizard or directly in the binding file. See "Defining a Flavor of SQL" on page 9 for details.

For complete information about specifying this and other parameters in the application connect string, see *Attunity Connect Data Adapter Guide and Reference*.

Index

A	12
ABS function	GROUP_SHOULD_AGG syntax file function 12
in syntax file 14	
AGG_DIST_NOT_SUPPORT syntax file func-	1
tion 11	IFNULL function
ASCII function	in syntax file 14
in syntax file 14	IN function
	in syntax file 14
С	III Syficax file 11
CASE function	J
in syntax file 14	JOLT
CASE_SENSITIVE syntax file function 11 CEIL function	syntax 6
in syntax file 14	
CHR function	L
in syntax file 14	LENGTH function
CONCAT () function	in syntax file 14
in syntax file 14	LIKE function
concatenation	in syntax file 14
in syntax file 14	LOG function
connect string	in syntax file 14
syntax parameter 20	LOWER function
CONVERT function	in syntax file 14
in syntax file 14	LPAD function
<i>y</i>	in syntax file 14
E	LTRIM function
_	in syntax file 14
Excel	
syntax 6	M
EXP function	MOD function
in syntax file 14	in syntax file 14
EXPR_IN_ORDERBY syntax file function 11	III Sylicax life 14
_	N.I.
F	N
FLOOR function	NAV.SYN
in syntax file 14	ABS function 14
	AGG_DIST_NOT_SUPPORT 11
G	ASCII function 14
GROUP_ANY_EXPR syntax file function 12	CASE function 14
GROUP_BY_ALIAS syntax file function 12	CASE_SENSITIVE 11
GROUP_BY_COLUMN syntax file function 12	CEIL function 14
GROUP_EXPR_BY_NUM syntax file function	CHR function 14
GIVO OI _LIM IV_DI_IVOM Syntax interaction	CONCAT () function 14

CONVERT function 14	SQL/MX 6
errors 9	SQRT 14
Excel 6	SUBSTR function 14
EXP function 14	supported functionality 11
EXPR_IN_ORDERBY 11	trigonometry functions 14
FLOOR function 14	TRUNC function 14
function variations 14	NO_COLUMN_ALIAS syntax file function 12
GROUP_ANY_EXPR 12	NO_DISTINCT_HAVING syntax file function 12
GROUP_BY_ALIAS 12	NO_EXPRESSIONS_IN_INSERT syntax file
GROUP_BY_COLUMN 12	function 12
GROUP_EXPR_BY_NUM 12	NO_GRANT syntax file function 12
GROUP_SHOULD_AGG 12	NO_MULTI_THREADING syntax file function
IFNULL function 14	12
IN function 14	NO_OWNER syntax file function 12
JOLT 6	NO_PARAMETERS_IN_SUBQUERY syntax
LENGTH function 14	file function 12, 13
LIKE function 14	NO_TWO_PARAMS_IN_COMPARE syntax file
LOG function 14	function 13
LOWER function 14	NO_TWO_PARAMS_IN_MATH syntax file func-
LPAD function 14	tion 13
LTRIM function 14	
MOD function 14	0
NO_COLUMN_ALIAS 12	ONE_COLUMN_SUBQUERY syntax file func-
NO_DISTINCT_HAVING 12	tion 13
NO_EXPRESSIONS_IN_INSERT 12	ORDER_BY_COLUMN syntax file function 13
NO_GRANT 12	ORDER_EXPR_BY_NUM syntax file function
NO_MULTI_THREADING 12	13
NO_OWNER 12	ORDER_NO_CALC syntax file function 13
NO_PARAMETERS_IN_SUBQUERY 12,	onzeneros syman meramon re
13	D
NO_TWO_PARAMS_IN_COMPARE 13	P
NO_TWO_PARAMS_IN_MATH 13	PARAM_AND_EXPR_IS_PARAM syntax file
ONE_COLUMN_SUBQUERY 13	function 13
ORDER_BY_COLUMN 13	PI function
ORDER_EXPR_BY_NUM 13	in syntax file 14
ORDER_NO_CALC 13	POSITION function
PARAM_AND_EXPR_IS_PARAM 13	in syntax file 14
PI function 14	POWER function
POSITION function 14	in syntax file 14
POWER function 14	
Rdb 6	R
ROUND function 14 RPAD function 14	Rdb
RTRIM function 14	version 5 syntax 6
SIGN function 14	ROUND function
single quotes 10	in syntax file 14
specifying variations 10	RPAD function
specifying variations to	in cuntay file 14

RTRIM function	JOLT 6
in syntax file 14	LENGTH function 14
	LIKE function 14
S	LOG function 14
_	LOWER function 14
SIGN function	LPAD function 14
in syntax file 14	LTRIM function 14
single quotes	MOD function 14
syntax file 10	NO_COLUMN_ALIAS 12
SQL	NO_DISTINCT_HAVING 12
adapting nonstandard 9, 18	NO_EXPRESSIONS_IN_INSERT 12
Attunity Connect 19	NO_GRANT 12
functions 14	NO_MULTI_THREADING 12
supported functionality 11	NO_OWNER 12
variations 10	NO_PARAMETERS_IN_SUBQUERY 12
SQL functions	13
in syntax file 14	NO_TWO_PARAMS_IN_COMPARE 13
SQL statement variations	NO_TWO_PARAMS_IN_MATH 13
syntax file 10	ONE_COLUMN_SUBQUERY 13
SQL/MX	ORDER_BY_COLUMN 13
syntax 6	ORDER_EXPR_BY_NUM 13
SQRT function	ORDER_NO_CALC 13
in syntax file 14	PARAM_AND_EXPR_IS_PARAM 13
SUBSTR function	PI function 14
in syntax file 14	POSITION function 14
syntax file	POWER function 14
ABS function 14	Rdb 6
AGG_DIST_NOT_SUPPORT 11	ROUND function 14
ASCII function 14	RPAD function 14
CASE function 14	RTRIM function 14
CASE_SENSITIVE 11	settings in connect string 20
CEIL function 14	SIGN function 14
CHR function 14	single quotes 10
CONCAT () function 14	specifying variations 10
CONVERT function 14	SQL/MX 6
errors 9	SQRT 14
Excel 6	SUBSTR function 14
EXP function 14	supported functionality 11
EXPR_IN_ORDERBY 11	trigonometry functions 14
FLOOR function 14	TRUNC function 14
function variations 14	syntax parameter 20
GROUP_ANY_EXPR 12	-J F
GROUP_BY_ALIAS 12	-
GROUP_BY_COLUMN 12	Т
GROUP_EXPR_BY_NUM 12	trigonometry functions
GROUP_SHOULD_AGG 12	in syntax file 14
IFNULL function 14	TRUNC function
IN function 14	in syntax file 14

variations in SQL functionality in syntax file 11